AD-764 073

THE IMAGE PROCESSING DATA MANAGER

Robert E. Nimensky, et al

System Development Corporation

Prepared for:

Advanced Research Projects Agency
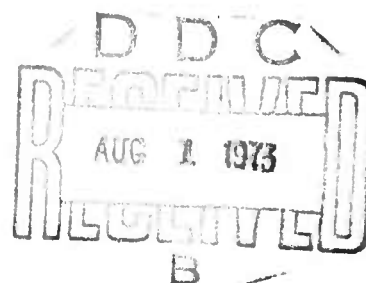
22 June 1973

SYSTEM DEVELOPMENT CORPORATION

THE IMAGE PROCESSING DATA MANAGER

R. E. NIMENSKY

JOAN BEBB

22 JUNE 1973
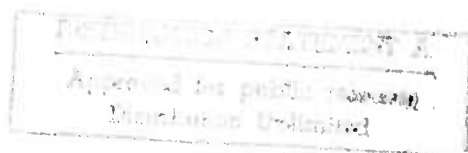
TM-5068/003/00

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
System Development Corporation 2500 Colorado Avenue Santa Monica, California 90406	2b. GROUP

3. REPORT TITLE

The Image Processing Data Manager

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Special Technical

5. AUTHOR(S) *(First name, middle initial, last name)*

Joan E. Bebb, Robert E. Nimensky

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
22 June 1973	30 32	--

8a. CONTRACT OR GRANT NO. DAHC15-73-C-0080	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO. ARPA Order No. 2254, Program Code 3D30	TM-5068/003/00
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	ARPA/Information Processing Techniques

13. ABSTRACT

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT

SYSTEM DEVELOPMENT CORPORATION

THE IMAGE PROCESSING DATA MANAGER

R. E. NIMENSKY

JOAN BEBB

22 JUNE 1973

TM-5068/003/00

ii

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'd)

ILLUSTRATIONS

1.   INTRODUCTION

This document describes a model of an image processing data management system called the Image Processing Data Manager. The Data Manager is an interactive program which, using a data base, aids the user in performing selective image processing. It answers his queries and allows him to specify conditional process operation and data retrieval. The user communicates with the Data Manager using a language which is a combination of English and algebra. The system has an underlying formal syntactical language (which may be used), but also incorporates a set of flexibility-introducing techniques which allow natural, English-like communication.

The Data Manager is implemented for the IBM/370 using the ICOS time-sharing system and the CWIC meta-compiler languages. It is readily transferable to operation under either IBM TSO or OS.

The data base associated with the Data Manager is a catalog of what is available to the image processor in terms of resource images; processed pictures; processing programs; and other data sets, such as specialized or general transformation matrices. It does not contain the actual image data or programs. Descriptions of processing techniques, individual program parameter descriptions, and job creation information may also reside in it. The processing programs or systems described need not be an integral part of the Data Manager; they may, in fact, operate on a different computer under a foreign operating system. The Data Manager creates jobs, as a user might. The current test data base is structured to create jobs for the IBM 370 Operating System and to use specific image processing programs which operate on these machines (VICAR*, for example).

The Data Manager can be used as an experimental tool for: (1) studying the language requirements of a user-oriented query language for image processing, (2) studying the data structure necessary for efficient storage and retrieval of pertinent information about images, and (3) learning how to organize a data base and library of programs so that all the parameters necessary for program execution can be extracted from the data base.

2.   THE HIERARCHICAL DATA BASE

To the Data Manager, an image is an object; objects belong to one or more abstract classes of objects. Thus, in the hierarchy in Figure 1, all the objects, M1, M2, V1, V2,..., L3, belong to the class ARRAY; M1 also belongs to the sub-classes PICTURE and MARINER.

A specific object is described by its attributes. Attributes are delineated in attribute-value pairs. For example, ROW=1000 is an attribute-value pair describing how many rows of pixels there are in picture M1.
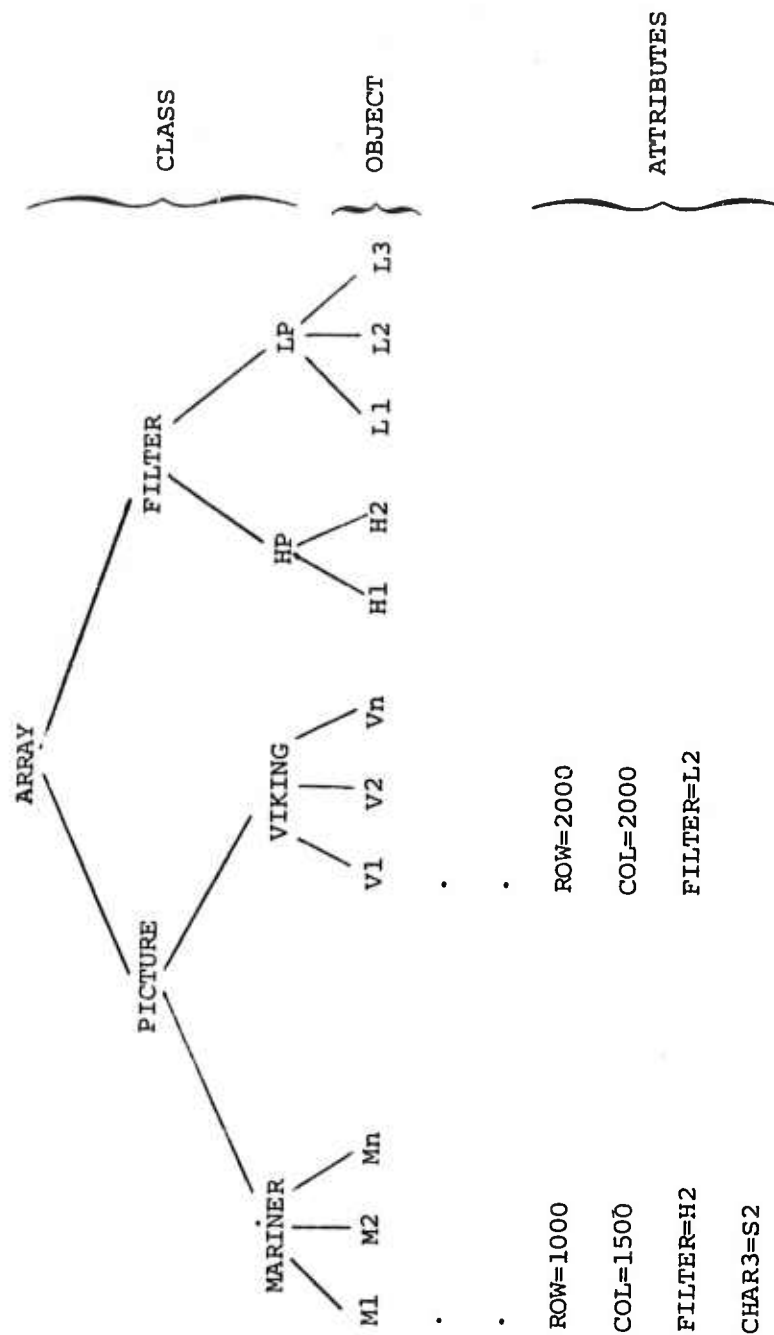
1

*A generalized image enhancement system developed by Jet Propulsion Laboratories, Pasadena, California.

Figure 1. Hierarchical Structure

2

These hierarchical constructs are the basis for retrieving data from the data base. Group, object, and attribute relationships can be expressed in simple data-base creation statements. To define the class structure expressed in Figure 1:

```
ARRAY: PICTURE, FILTER;
PICTURE: MARINER, VIKING;
FILTER: HP,LP;
MARINER: M1,M2,M3;
VIKING: V1,V2,V3;
HP: H1,H2;
LP: L1,L2,L3;
```

Attribute-value pairs are similarly defined:

```
M1:  ROW:=1000,COL:=1000,FILTER:=H2, CHAR3:=S2;
V1:  ROW:=2000,COL:=2000,FILTER:=L2;
```

Very complex data relationships can be expressed with these forms. They are a flexible list structure which allows members of a list to point to other lists. With the attribute-value pairs, FILTER and CHAR3, the values are other lists, which may in turn point to still other lists. Note that attributes may be physical descriptors (such as ROW) or processing indicators (such as FILTER).

3. THE QUERY LANGUAGE

Numerous facilities of the Data Manager are provided through semantic processing, rather than through syntactic analysis. That is, the definition of an item controls the interpretation of a phrase rather than the item's placement within the phrase.

3.1 FLEXIBILITY

The Data Manager's query language is English-oriented. Most questions are a combination of English and mathematical notation. A synonym capability allows a user to specify many spellings of a word. For instance, the attribute COL may also be specified as COLUMN, COLUMNS, or COLS (if these words are defined as synonyms for COL). Noise words (words which are ignored by the Data Manager) can also be defined by the user to make queries more natural. Consider an example. To obtain a list of pictures which are in a given region, along with their coordinates, the user may give a query and an answer directive using a formal statement:

PIC REGIONA=>C ORD

He may, however, use synonyms and noise words. This same query can be made as:

FOR ALL THE PICTURES IN REGIONA LIST THEIR COORDINATES

3

3.2   POWER

The query language has a powerful boolean expression capability to suit a wide range of potential users. The simple statement above may involve a complex boolean expression, REGIONA, which determines if the coordinates of a picture lie within a given region:

    REGIONA: R1 OR R2 OR R3 OR R4 OR R5

REGIONA is an R1 or an R2... or an R5. R1 is a boolean expression, as are R2, R3, R4, and R5. Each Rn expression is a comparison of a coordinate of a picture in the data base with the coordinates of a given region. Each picture is assumed to have five such coordinates specified--the corners and the center. If any of the Rn's is true, the REGIONA expression is true; the picture is in the region.

A possible R1 expression is:

    R1: 300000<LT1<700000 AND 1400000<LG1<1800000

LT1 and LG1 are the first latitude and longitude coordinates of the picture being tested. R1 demands that the first latitude must be greater than 30° 00' 00" and less than 70° 00' 00" and that the first longitude must be greater than 140° 00' 00" and less than 180° 00' 00". R2 through R5 may make similar conditions on other picture reference coordinates.

Once written, these expressions may be saved in the data base; any user may now exercise the query (or a variation of it) with little awareness of the specifics of the expressions.

4.   REMOTE PROCESSING

One of the most powerful features of the Data Manager is its ability to obtain program execution parameters directly from the data base. It can, as one of its options, generate a complete image processing job (to be executed external to the Data Manager). As a model, the existing implementation of the Data Manager creates VICAR jobs.

The Data Manager finds all the data base entries that meet the conditions of a query. For the query concerning pictures in REGIONA, every picture which has at least one latitude and longitude coordinate in the defined region qualifies. The list of entries is passed on to the answer processor. If job creation is indicated in the answer portion:

    PICTURES IN REGIONA=>MOSAIC

and MOSAIC is a job, the job processor obtains the information required for job formation from the data base itself. This information includes the

4

locations of data sets, their file numbers, and the programs that are to be
executed and their parameter requirements. In this way the Data Manager serves
as a front-end processor for foreign image processing systems.

5.   DATA-BASE CREATION STATEMENTS

There are four data base creation (or amplification) statements. With them
the user can create hierarchical data structures, assign attributes to objects
or groups and provide synonyms for his own defined terms or for system operators.
These statements are permissible at any time during operation of the Data
Manager. There is no unique data-base-generation mode; at any time the Data
Manager operates on its current information, permitting modification or
extension.

5.1   OBJECT-CREATION STATEMENT

Form:

        Object name: OBJ:= identifier [, attribute-pair]

An object has a referent in the real world such as a picture, a camera, or
a lens. An object statement identifies this entity to the system.

Attributes describe properties of an object and are associated with the object
as attribute-pair descriptors. An object may have any number of attribute pairs.
An attribute pair consists of an attribute name (which identifies the attri-
bute) and an attribute value. The attribute value may be an identifier (which
is interpreted as a pointer to an object), a number, a quoted string of
characters, or a list of these values.

$$\text{Attribute name:=} \begin{bmatrix} \text{number} \\ \text{text} \\ \text{identifier} \\ \text{list} \end{bmatrix}$$

For example, to describe the picture, M1, the following object-creation state-
ment could be used:

        M1: OBJ:= ·M1, ROW:= 1000, COL:= 1000,

            FILTER:= (F2, F3, F4), COMMENT:= 'SLOT IS A DEFECT IN ENS'

The picture is named M1; its attributes are ROW, COL, FILTER, and COM   T.
Attribute values are expressed as numbers (1000), a list of identifiers/pointers
(F2, F3, F4), and a text string.

An object must have an OBJ attribute to identify it as such. The identifier associated is usually the same as the object name but may be any name (as will be seen in the section describing the creation of jobs for remote processing).

5.2   GROUP STATEMENT

Form:

        Group name: identifier, identifier [,identifier]

The group statement collects a list of identifiers and assigns the group attribute to the list. This statement is used to create hierarchical data structures; the list of identifiers may include names of groups or objects.

To build the following structure:
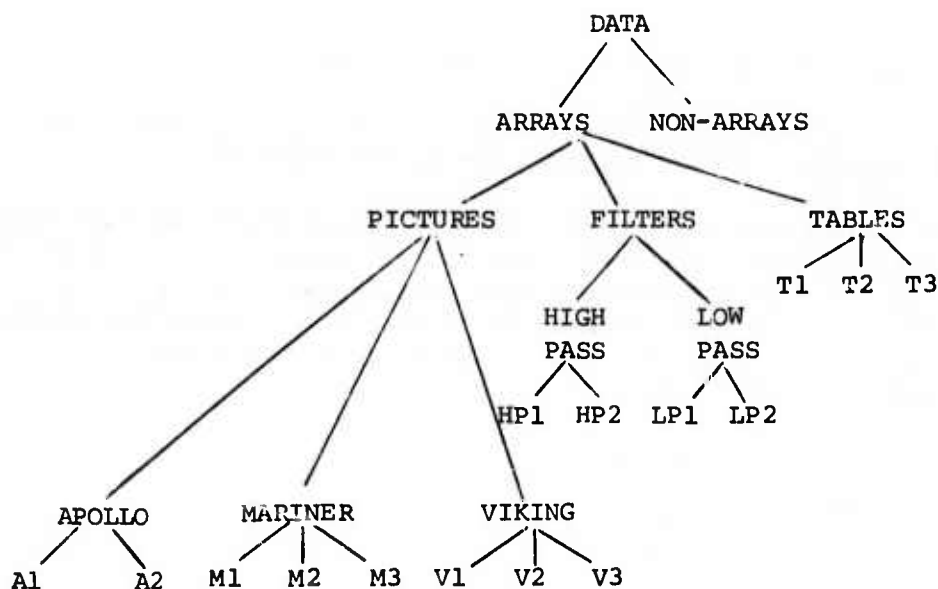
the user issues the following group statement:

```
DATA:        ARRAYS, NON-ARRAYS
ARRAYS:      PICTURES, FILTERS, TABLES
PICTURES:    MARINER, VIKING, APOLLO
FILTERS:     HIGH-PASS, LOW-PASS
MARINER:     M1, M2, M3
VIKING:      V1, V2, V3
APOLLO:      A1, A2
HIGH-PASS:   HP1, HP2
LOW-PASS:    LP1, LP2
TABLES:      T1, T2, T3
```

6

5.3   GROUP ATTRIBUTE QUALIFIERS

Form:

Group-statement, attribute-name:= attribute value

[, attribute-name:= attribute-value]

Attribute pairs, consisting of an attribute name and an attribute value, can be added to any group statement. These attributes apply to all groups and objects contained in the group unless they are overridden in a lower (less inclusive) attribute assignment. Specifically, an attribute such as date may be applied to a whole set of entities by assigning the date at the group level. Or, if all MARINER pictures have been acquired using the same scanner, the scanner attribute can be attached at the group level, rather than to each individual picture. If, however, a few of the MARINER pictures were scanned with a different device, these few pictures can have that attribute attached to the individual pictures, overriding the group attribute.

For example:

MARINER: M1, M2, M3, SCANNER:= SC32M,

DATE:= [10, 11, 73]

The group consists of M1, M2 and M3; the SCANNER and DATE attributes have been added.

5.4   SYNONYMS

Form:

identifier: identifier [: identifier]

5.4.1   <u>Synonyms for Groups, Objects, and Attributes</u>

The synonym statement enumerates synonyms for the name of an object, group or attribute. <u>One</u> and <u>only one</u> identifier in the list must have been used to create a group, object, or attribute. All other identifiers are stored in the data base with:

SYN:= identifier

The identifier is the previously known item. When a word with a SYN attribute is encountered, the identifier's value is substituted.

For example:

COL: COLS: COLUMNS: COLUMN

If COL is previously defined, COLS, COLUMNS, and COLUMN each acquire a SYN attribute which points to COL. When COLS, COLUMN, or COLUMNS is used, COL is substituted.

5.4.2   Synonyms for Operators

Form:

Identifier: OP:= identifier, RELOP:= known-identifier

Words may be made synonymous with some of the special symbols the system uses to signal relational operators. This is done by using the form above, where the identifier is the user's word, the known identifier is the system mnemonic for the particular relational symbol, and RELOP and OP are the key attributes necessary for synonymous operator definition.

The symbolic operators and equivalent system mnemonics for operators that can be synonymously extended are:

    =   EQ
    <   LS
    >   GR

For example:

EQUAL: OP:= EQUAL, RELOP:= EQ

INPUT PHASE   ROW EQUAL COL

EQUIVALENCE: ROW = COL

5.5   USER-DEFINED EXPRESSIONS

As part of data base creation, the user may define arithmetic or boolean expressions to be used as part of subsequent user query statements.

5.5.1   Arithmetic Expressions

Form:

label: operand operator operand

An arithmetic expression specifies a value in a way similar to ordinary algebra.

8

It is a labeled combination of operands and operators. The label is any combination of letters and digits beginning with a letter.

5.5.1.1 Operands

The operands of an arithmetic expression may be numeric-valued attributes, constants, or previously defined arithmetic expressions. An attribute used as a value in an arithmetic expression may appear by itself or in combination with an object or group. If it appears by itself, it will take on the value of the attribute for each object being evaluated; if it is paired with an object, it is a single value; if it is paired with a group, it takes on the value of the attribute for group members. An arithmetic expression is evaluated for each object in the range of the expression.

For example:

    DIM: ROW * COL

    PICTURE DIM

In this example, the arithmetic expression named DIM contains two attributes: ROW and COL. When PICTURE and DIM are paired to form another expression, every object in the PICTURE group has its rows multiplied by its columns to evaluate the operand.

Within arithmetic expressions, constants may be written either in standard mathematical notation or in linearized power-of-10 notation:

    10.35   +11.3   -18.65   500

    .365E-4   .4987E+7   328

In the above example, E-4 indicates multiplication by 10^{-4}.

5.5.1.2 Operators

Arithmetic operators are used to combine operands into arithmetic expressions. The arithmetic operators are:

+	addition	ARCSIN($\cdot$)	arcsine
-	subtraction	ARCCOS($\cdot$)	arccosine
*	multiplication	LN($\cdot$)	natural logarithm
/	division	ATOBX(a,b)	a^b
ABS($\cdot$)	absolute value	MAX($\cdot$)	maximum
SIN($\cdot$)	sine	MIN($\cdot$)	minimum
COS($\cdot$)	cosine		

Within arithmetic expressions, multiplication must be explicitly stated.

5.5.2   Boolean Expressions

A boolean expression is one which evaluates to one of two values: true or
false. An arithmetic expression is a special case of a boolean in that it is
true if the attributes in the expression exist for the entity being tested.

5.5.2.1   Relational Expressions

Form:

        Label: operand relational-operator operand

where each operand is an arithmetic expression or the name of an arithmetic
expression and the relational operators are:

        =   equal
        /=   not equal
        <   less than
        >   greater than
        <=   less than or equal
        >=   greater than or equal

For example:

        RECTANGLE: ROWS/=COLS

        SQUARE: ROWS=COLS

5.5.2.2   Logical Expressions

Form:

        Label: operand logical-operator operand

where each operand is a relational expression or the name of a relational
expression and the logical operators are:

        AND

        OR

For example:

        R1:   LAT>100 AND LONG>175

        BIG-RECTANGLE: RECTANGLE AND DIMENSION>5000000.

In the first example, two relational expressions are combined. In the second,
BIG-RECTANGLE is defined as a RECTANGLE (a previously defined expression) and
a new relational expression (using a previously defined arithmetic expression).

10

6.   QUERY STATEMENTS

A query to the Data Manager is stated in the form of a conditional function. The
function consists of two parts: the condition portion and the answer portion.
The condition portion is a boolean expression which, in use, produces a list
of names of data base entries satisfying the condition of the expression. The
answer portion (optional) structures the Data Manager's response. It defines
a series of arithmetic expressions which are evaluated; a value for every
member of the list is returned.

        boolean-expression [=>arithmetic expression [,arithmetic expression]]

The answer portion is signaled by the arrow (=>). The word LIST or PRINT may
be used in place of the arrow to serve the same function.

If a query consists of only a boolean expression, the names of those objects
satisfying the expression are returned (along with the last attribute of
the expression).

For example: A query

        PICTURE ROW > 2015

returns:

PICTURE	ROW
M2	2300
M5	3015
M6	2016

Since an arithmetic expression may be simply the name of an attribute; the
answer portion can contain a list of attributes to be printed for each hit.

For example:

        PICTURE   ROW>1050 => COL, DIM

would cause to be printed

PICTURE	ROW	COL	DIM
M1	1100	1100	1210000
M2	2300	2000	4600000
M5	3015	3000	9045000
M6	2016	2000	2016000

6.1   The KNOW Attribute

If the user desires to know all the attributes of a particular object, he can use the special KNOW attribute.

For example:

        P3 KNOW

will return:

ATTRIBUTE	VALUE
OBJ	P3
ROW	100
COL	100
OWNER	ROB NEMO
DATE	12
	15
	73
FILTER	F5

6.2   Pointer Attributes

The values of certain attributes may be pointers to other objects. In the example above, F5 is another object. Without knowing the value of FILTER (F5), FILTER can be indirectly addressed and F5's attributes retrieved.

For example:

        P3 FILTER ROW

returns the value of the ROW attribut. of F5, not the ROW attribute of P3.

Pointer attributes may be added indefinitely to a group or object as long as the previous attribute is a pointer attribute.

For example:

        P3 CAMERA LENS APERTURE

This attribute search finds the CAMERA attribute of P3, then the LENS of that CAMERA and finally the APERTURE of that LENS.

6.3 REVERSAL OPERATORS

It is more natural to express the query in the lens aperture example in reverse order. The words OF or IN may be used to reverse the order of any pair.

For example:

APERTURE OF LENS OF CAMERA OF P3

is equivalent to:

P3 CAMERA LENS APERTURE

However, since some natural usages of the word IN maintain the sequence of 'object attribute' or 'object condition' rather than reverse it, such as in the phrase:

PICTURES IN REGIONA

where REGIONA is a boolean expression, the Data Manager performs reversal only if the first element of the triple:

WORD IN WORD

is not an object or a group.

6.4 NOISE WORDS

To make queries more English-like and more natural, the user may extend his query vocabulary by defining a set of noise words. This is done by assigning the NOISE attribute to these words. Noise words are ignored.

To define noise words, create an object with a NOISE attribute and then a list of synonyms for that object.

For example:

FOR: NOISE:= NOISE

FOR: ALL: WHAT: IS: THE: USED

Now the aperture previously asked for can be requested by:

WHAT IS THE APERTURE OF THE LENS IN THE CAMERA USED IN P3.

7. REMOTE PROCESSING

The answer to a query might be the creation of a computer job to be executed
subsequently. This job may call on programs distinct from the Data Manager,
but available within the total system. The VICAR image processing sub-system
comprises such a set of programs. Through its facility to generate jobs for
remote processing, the Data Manager can be made to serve as a front-end data
management system for an image processing laboratory.

7.1 PREPARING THE DATA BASE

The Data Manager generates the job control language (JCL) and all parameters
required for the job to be executed. The data base creator stores the defini-
tion of the JCL and parameters in the data base. The JCL may be added to or
changed by modifying the data base; there is no need to reprogram the JCL
generation program. The subsequent sections describe how the JCL is stored in
the data base using standard data-base-creation statements. Use of the JCL
attribute is essential to this process.

Consider a job called MOSAIC, which produces a mosaic of all the pictures in a
given region. Each picture will be projected by the program PROJECT and then
all of the projected images will be input to the MOSAIC program, which produces
the final mosaic. The region is defined by the latitude and longitude coordinates
of a rectangle which encompasses the area. A picture is considered in the region
if any of the reference points in the picture have latitude and longitude
coordinates which lie within the rectangle. Query interpretation supplies a
list of pictures falling in the region; the answer processor creates the job.

7.1.1 Defining the Objects and Condition

Data base entries for each picture follow the pattern of two samples, P1
and P2.

```
P1:OBJ:=LT1:=605000,LG1:=1474500,LT2:=545000,LG2:=1481500,
       LT3:=615500,LG3:=1533300,LT4:=550000,LG4:=1541200,
       LT5:=581000,LG5:=1510000,ROW:=775,COL:=900,
       FORMAT:=FORMATB,VOL:='SER=TAPE1',FILE:=3,ID:=TAPE1;
P2:OBJ:=P2,LT1:=544000,LG1:=1541000,LT2:=533400,LG2:=1545900,
       LT3:=555200,LG3:=1595200,LT4:=535400,LG4:=1605800,
       LT5:=520600,LG5:=1573500,ROW:=775,COL:=900,
       FORMAT:=FORMATB,VOL:='SER=TAPE2',FILE:=6,ID:=TAPE2;
```

The LT and LG attributes are coded names for five reference latitude and
longitude coordinates for each picture (the corners and the center). The
FORMATB attribute specifies the JCL parameters for tapes, VOL is another
JCL parameter (tape volume), FILE is the file number on the particular tape,
and ID is the name of a tape.

14

A PICTURE or PICTURES (synonym) is defined as a group:

> PICTURE:P1,P2,P3,P4,P5,P6,P7;

A REGION is defined to be the boolean expression:

> REGION:R1 OR R2 OR R3 OR R4 OR R5;

R1 through R5 are definitions of coordinate relationships for each of the five coordinates of a picture. The first three are shown below:

```
R1:(EXTENT COORD LT2<LT1<EXTENT COORD LT1) AND
   (EXTENT COORD LG2<LG1<EXTENT COORD LG3;
R2:(EXTENT COORD LT2<LT2<EXTENT COORD LT1) AND
   (EXTENT COORD LG2<LG1<EXTENT COORD LG3);
R3:(EXTENT COORD LT2<LT3<EXTENT COORD LT1) AND
   (EXTENT COORD LG2<LG3<EXTENT COORD LG3);
EXTENT:OBJ:=EXTENT, COORD:=PROJ1;
PROJ1:TYPE:=MERCATOR,LT1:=700000,LG1:=1400000
    ,LT2:=300000,LG2:=1400000
    ,LT3:=700000,LG3:=1800000
    ,LT4:=300000,LG4:=1800000;
```

7.1.2   Defining the Job

The following object creation statement sets up the MOSAIC job.

> MOSAIC: VICAR:=MOSAIC, JCL:=VSETUP,PGM:=(PROJECT,MOSAIC2),OUTPUT:=
> (SYSTEM, USER, OUTNAME)

The job uses components of the VICAR system (PROJECT and MOSAIC2), the JCL list is called VSETUP (and is, in this example, directed to the IBM OS system), and the OUTPUT is specified indirectly. As a convention, a listed value for certain attributes signals that the value is to be interpreted indirectly (just as with the range in a boolean expression). In this case, the OUTPUT specification is found by looking at the SYSTEM entry. SYSTEM's USER attribute points to a specific user. USER's value points to another attribute which finally determines the output file description. This indirect addressing built into the job description allows each user to define his own output destination without generating a separate MOSAIC job description.

For example:

> SYSTEM:USER:=NIMENSKY;
>
> NIMENSKY:JOBID:='(08648,98312,3325)',OUTNAME:=BOBOUT;
> BOBOUT:FORMAT:=FORMATB,VOL:='SER=BOBOUT',ID:=BOBOUT;

are the data base entries which specify a particular user, NIMENSKY. The Data
Manager knows how to locate all the variable data in the job description,
relieving the user of the necessity of defining data over and over again. The
Data Manager goes from SYSTEM to NIMENSKY to find that OUTNAME is BOBOUT; if
another user were operating the system, he could change SYSTEM: USER to his
name (in his copy of the system) so that his JOBID and OUTNAME would be used
in the generated JCL.

For every VICAR job, there are eight standard JCL statements. These are defined
in VSETUP, the data base entry pointed to in the MOSAIC definition.

```
VSETUP:VICRUNX,JOBLIB,STEP1,SYSOUT,SYSUDUMP,FT06F001,FT07F001,VSYS00;
VICRUNX:JCL:=JOB,BLANK:=(SYSTEM,USER,JOBID),BLANK:=
        (SYSTEM,USER),CLASS:=D,TIME:=15;
JOBLIB:JCL:=DD,DSN:='IPL1.VICARLIB',DISP:=SHR,VOL:='SER=000530',
        UNIT:=2314;
STEP1:JCL:=EXEC,PGM:=VMAST,REGION:='250K',TIME:=15;
SYSOUT:JCL:=DD,SYSOUT:='A',DCB:='BLKSIZE=3520';
SYSUDUMP:JCL:=DD,SYSOUT:='A';
FT06F001:JCL:=DD,SYSOUT:='A';
FT07F001:JCL:=DD,SYSOUT:='B';
VSYS00:JCL:=DD,UNIT:=2314,SPACE:='(TRK,2)';
```

Other jobs using programs from other subsystems generate different JCL;
the job definition need only refer to a different list. In this case, the
first JCL statement, VICRUNX, is a JOB statement (modeled after SDC's standard
job card). The attribute pairs coincide with the keyword and value orientation
of the OS JCL format. The translation from data base entry to specific JCL
statement is primarily one of editing and interpretation of indirect addressing.
Within the Data Manager, a convention has been instituted to take care of unique
JCL formats. For instance, in the OS JOB statement, there is no keyword for
the job identification field. Within the Data Manager, the BLANK attribute is used
to indicate 'no keyword'. Since quoted textual strings are legal attribute
values, the SYSTEM USER JOBID is entered as JOBID:='(08648,98312,3325)' in the
NIMENSKY entry. The general rule for JCL generation is to use the name of the
statement as the statement label, ignore the 'JCL' but print its value as the
statement type, and then using the remaining attribute pairs, print the keyword,
an equal sign (=), and the value. Separate all successive pairs by commas and
adjust for end of line. The following JCL will be generated from the VSETUP
information when the MOSAIC job is exercised.

```
//VICRUNX    JOB  (08648,98312,3325),NIMENSKY,CLASS=D,TIME=15
//JOBLIB     DD   DSN=IPL1,VICARLIB,DISP=SHR,VOL=SER=000530,UNIT=2314
//STEP1      EXEC PGM=VMAST,REGION=250K,TIME=15
//SYSOUT     DD   SYSOUT=A,DCB=BLKSIZE=3520
//SYSUDUMP   DD   SYSOUT=A
//FT06F001   DD   SYSOUT=A
//FT07F001   DD   SYSOUT=B
//VSYS00     DD   UNIT=2314,SPACE=(TRK,2)
```

Next, for each data set that lies within the region, a data definition (DD)
statement must be generated. Because PROJECT has the REPEAT attribute, a
temporary data set will be established for each output picture. The basic JCL
for tape units (input to PROJECT) is generated from the FORMATB statement; the
temporary data sets are directed to disk and are generated from FORMATA. The
data base statements for this JCL follow:

```
PROJECT:PGM:=PROJECT,INPUT:='@',OUTPUT:=TEMP,PARAMS:=(PARAM1,
              PARAM2),REPEAT:='@',OUTSIZE:=(1,1,2000,2000);
PARAM1:PARAMS:=(EXTENT,COORD);
PARAM2:PARAMS:=('@',COORD);
MOSAIC2:PGM:=MOSAIC2,INPUT:=(TEMP),OUTPUT:=(SYSTEM,USER,OUTNAME);
FORMATA:JCL:=DD,UNIT:=2314,DCB:='(BLKSIZE=6000,LRECL=2000)',
              SPACE:='(6000,(0710),,,ROUND)';
FORMATB:JCL:=DD,LABEL:='(,BLP)',DISP:='(OLD,KEEP)',UNIT:=TAPE7,
              DCB:='(BLKSIZE=0,LRECL=0,DEN=2,TRTCH=C)',VOL:=('@',VOL);
```

The generated JCL will be:

```
//VSYS01     DD      LABEL=(,BLP),DISP=(OLD,KEEP),UNIT=TAPE7,
//       DCB=(BLKSIZE=0,LRECL=0,DEN=2,TRTCH=C),VOL-SER-TAPE1
//VSYS02     DD      LABEL=(,BLP),DISP=(OLD,KEEP),UNIT=TAPE7,
//       DCB=(BLKSIZE=0,LRECL=0,DEN=2,TRTCH=C),VOL=SER=TAPE2
//VSYS03     DD      LABEL=(,BLP),DISP=(OLD,KEEP),UNIT=TAPE7,
//       DCB=(BLKSIZE=0,LRECL=0,DEN=2,TRTCH=C),VOL-SER-TAPE3
//VSYS04     DD      LABEL=(,BLP),DISP=(OLD,KEEP),UNIT=TAPE7,
//       DCB=(BLKSIZE=0,LRECL=0,DEN=2,TRTCH=C),VOL=SER=TAPE4
//VSYS05     DD      UNIT=2314,DCB=(BLKSIZE=6000,LRECL=2000),
//       SPACE=(6000,(0710),,,ROUND),DSN=&&S5
//VSYS06     DD      UNIT=2314,DCB=(BLKSIZE=6000,LRECL=2000),
//       SPACE=(6000,(0710),,,ROUND)DSN=&&S6
//VSY07      DD      UNIT=2314,DCB=(BLKSIZE=6000,LRECL=2000),
//       SPACE=(6000,(0710),,,ROUND),DSN=&&S7
//VSYS08     DD      UNIT=2314,DCB=(BLKSIZE=6000,LRECL=2000),
//       SPACE=(6000,(0710),,,ROUND),DSN=&&S8
//VSYS09     DD      LABEL=(,BLP),DISP=(OLD,KEEP),UNIT=TAPE7,
//       DCB=(BLKSIZE=0,LRECL=0,DEN=2,TRTCH=C),VOL=SER=BOBOUT
```

17

Setting up the actual job (which is a parameterized call to the VICAR
driver, VMAST) begins with the standard input (SYSIN) file definition. The
input files and output files are determined from the VICAR description. The
specific parameters are taken from the data base.

The parameters for PROJECT are described by its PARMS attribute: PARAM1,
PARAM2. Proceeding down another level, PARAM1 is defined as an indirect
address, EXTENT COORD--that is, the value of the COORD attribute of EXTENT.
The definition of EXTENT is:

```
            EXTENT:OBJ:=EXTENT,COORD:=PROJ1;
            PROJ1:TYPE:=MERCATOR,LT1:=700000,LGL:=1400000
                      ,LT2:=300000,LG2:=1400000
                      ,LT3:=700000,LG3:=1800000
                      ,LT4:=300000,LG4:=1800000;
```

The COORD of EXTENT is PROJ1. Thus, the actual set of parameters are the values
of PROJ1. When generating parameters, all the attribute names are ignored.
PARAM2 is defined using '@'. This signals an indirect reference to the input
data set: the coordinates of the input picture will be used for the second set
of parameters. A sample portion of the parameter generation for PROJECT is:

```
// SYSIN      DD      *,DCB=BLKSIZE=80
                           S1 FILE   3                    1
      1 PROJECT      VTESTX        1      1 2000 2000  1 0 0 0 0 0 0 0 0 5 0 0 0
   MERCATOR      70.00.00      140.00.00     30.00.00   140.00.00   70.00.00
  180.00.00      30.00.00     180.00.00
   60.50.00     147.45.00      54.50.00    148.15.00    61.55.00.  153.33.00
   55.00.00     154.12.00      58.10.00    151.00.00
```

The parameters and job control for the mosaicking step are generated similarly.
The complete data base is contained in Appendix A: a complete example of
JCL generation is in Appendix B. All the JCL is generated based upon
the particular data base entries satisfying the query conditions or upon the
data base definition of the job.

7.2   CREATING A JOB THROUGH QUERY

Once the proper data base entries have been configured, the user exercises the
system's job creation facility with a simple query:

            FOR ALL PICTURES IN REGIONA =>MOSAIC

The system finds those pictures which lie within a particular region (REGIONA).
The range of all pictures is the group, PICTURE; REGIONA is a rather complex
boolean expression (given previously).

The query is parsed so that the noise words FOR and ALL are ignored and the reversal operator IN is recognized, but also ignored, because PICTURES, synonymous with PICTURE, is a group. Every member of PICTURE -- P1, P2, P3, P4, P5, P6, and P7 -- is tested against the boolean expression, REGIONA. If the boolean expression is true, the name of that picture is passed on to the MOSAIC processor.

Consider this evaluation as it proceeds for the first phrase in expression REGIONA: R1 (given previously). First, EXTENT COORD LT2 is evaluated. The value of COORD attribute of EXTENT is PROJ1. The LT2 attribute of PROJ1 is 30°00'00". The next phrase in R1 is <LT1<. LT1 is an attribute without an object; therefore the current object (picture) is used. The first picture, P1, has an LT1 of 60°50'00". The next phrase is EXTENT COORD LT1. The LT1 attribute of EXTENT is 70°00'00". The first part of R1 is evaluated as 30°00'00"<60°50'00"<70°00'00". Hence, the first coordinate of the first picture satisfies the expression; the picture P1 is put on the list of entities satisfying the query. Continued evaluation of all the pictures against the expression REGIONA produces a list of P1, P2, P3, and P4. Because MOSAIC (the desired response) has a VICAR attribute, the list of pictures is passed on for JCL generation.

The equations R1 through R5 could have been written incorporating the values of the coordinates, rather than using a complicated pointer (EXTENT COORD LT2 is a constant). Instead, EXTENT COORD's value is given as PROJ1, the coordinates of a particular region. There is a strong reason for the more involved specification. MOSAIC is a general process and REGIONA is a general boolean expression which could be used to find pictures in any region. If 50 regions were specified in the data base as PROJ1,...,PROJ50, the user need only set EXTENT COORD's value to the particular region he wants to process to MOSAIC any one of them. If constants were used, 50 versions of expressions R1 through R5 would have to be written, each with 20 constants, to accomplish the same facility.

Within the OS system, generating job control language on the appropriate data set will cause automatic execution of a job. Hence, the Data Manager may conditionally create and cause execution of a remote process using previously stored information in a very flexible manner.

8. <u>SYSTEM INTERACTION</u>

Once the Data Manager has been loaded, it asks the user to specify the location of his source (data base):

SOURCE=

The user responds with a file description, if he has a previously prepared card image file of data base creation statements and/or queries, or with

TERMINAL

if the data base is to be created from scratch. If a previously prepared file is signaled, the program reads this file, typing out any comments (see Section 9). On completion, it reverts to the user's terminal to request input. Its desire for input is signaled by an outputted asterisk (*).

During the course of using the system, the user may wish to amplify his data base from additional remote files or to reinitiate suppressed output. Such communication is achieved through a set of special commands of unique form. This form is a dollar sign ($), followed by a keyword. These commands are enumerated below and are permissible at any time that the program is requesting input by the asterisk prompt signal.

8.1 $SOURCE

This statement allows amplification of the data base or queries from remote files. The system will request a file description (as above) and read in the specified card image file.

8.2 $OUTPUT

This command restores the printing of output, if it was previously suppressed. That is, during the course of printing, the user is asked how many more lines of output are desired. If he responds NONE to this question, subsequent output is suppressed. He may restore printing the next time the system requests input by typing $OUTPUT.

8.3 $TREE

This is a debugging statement which causes the values input to the PRINTREE routine to be printed.

8.4 $UNTREE

This command turns off tree printing.

8.5 $BACK

This is a debugging statement which permits tracing of the Data Manager's execution in the case of a program error.

8.6 $NOBACK

This command suppresses tracing.

9.   #COMMENT

The comment following the # will be printed. This is a device available for generating printed comments while the Data Manager is loading a data base or test queries from a SOURCE file.

10.   SAVE

This command allows saving of the current program (data base). It stops the Data Manager in a restartable manner and returns control to the system monitor. The program and its current data base can now be named and saved. When the program is reloaded (by using its saved name), it will continue operation from the point at which it was saved. This command does not create a file re-readable using the SOURCE facility. Note that this command does not use the $ signal.

11.   SUMMARY

The Data Manager demonstrates the feasibility of a solution to the image processing data management problem. The hierarchical data base can be easily queried with an English-based language. The user can define his own arithmetic and boolean expressions (which may be saved in the data base). The job generation portion of the system demonstrates the feasibility of automatic retrieval of parameters from the data base and the direct execution of a complex task based upon a query statement with no further intervention by the user.

12.   FUTURE AMPLIFICATIONS

The Data Manager was designed, in part, to allow exploration of the possibility of semantic parsing of a query language. We will continue to develop the language so that meanings of words (as defined in the data base) determine the interpretation given a statement. The dependence on rigid syntax will be iteratively relaxed.

To further enhance the use of the Data Manager for applications other than image processing, a hierarchical attribute definition capability can be added.

To simplify user interaction with the data base, an assignment statement can be incorporated which allows the user to set a single attribute to a value or any number of attributes to a value based upon range and boolean condition.

The Data Manager can be made able to help the user find his way to the solution of a problem through incorporation of more refined "help" procedures than are now present.

The current model keeps its list structured data base in core memory. Obviously, this is very constraining. The problem of handling remote data storage must be solved. The model will be used to explore feasible ways of organizing such a large data base.

Finally, we plan to make many data bases of information from other disciplines to determine the applicability of the Data Manager concept to these areas.

APPENDIX A

SAMPLE DATA BASE

```
PICTURE:P1,P2,P3,P4,P5,P6,P7;
LT1:ATTRIB:=LT1; LT2:ATTRIB:=LT2; LT3:ATTRIB:=LT3; LT4:ATTRIB:=LT4;
LT4:ATTRIB:=LT4; LG1:ATTRIB:=LG1; LG2:ATTRIB:=LG2; LG3:ATTRIB:=LG3;
LG4:ATTRIB:=LG4; LG5:ATTRIB:=LG5;
R1:(EXTENT COORD LT2<LT1<EXTENT COORD LT1) AND
    (EXTENT COORD LG2<LG1<EXTENT COORD LG3);
R2:(EXTENT COORD LT2<LT2<EXTENT COORD LT1) AND
    (EXTENT COORD LG2<LG1<EXTENT COORD LG3);
R3:(EXTENT COORD LT2<LT3<EXTENT COORD LT1) AND
    (EXTENT COORD LG2<LG3<EXTENT COORD LG3)
DIM:ROW*COL;
AND:OP:=AND; OR:OP:=OR;
OF:REVOP:=IN,OP:=OF; ABOUT:REVOP:=IN,OP:=ABOUT; IN:OP:=IN,REVOP:=IN;
LIST:OP:=LIST; PRINT:OP:=PRINT;
PICTURE:PICTURES:PIC:IMAGE:IMAGES;
ROW:ATTRIB:=ROW;
ROW:ROWS:LINE:LINES;
COL:ATTRIB:=COL;
COL:COLUMN:SAMPLE:COLUMNS:SAMPLES:COLS;
DIM:DIMENSION:AREA;
FOR:NOISE:=NOISE;
ALL:FOR:THE:WITH:WHAT:THAN:A:HOW:MANY:DO:YOU:HAVE:ANY:OF;
FOR:THERE:WHERE:WHICH:IS:SHOW:ME;
GREATER:OP:=GREATER,RELOP:=GR;
EQUAL:OP:=EQUAL,RELOP:=EQ;
LESS:OP:=LESS,RELOP:=LS;
GREATER:LARGER:BIGGER:MORE;
LESS:SMALLER;
EQUAL:EQUALS:SAME;
FOR:AS:TO;
PROJ1:TYPE:=MERCATOR,LT1:=700000,LG1:=1400000
            ,LT2:=300000,LG2:=1400000
            ,LT3:=700000,LG3:=1300000
            ,LT4:=300000,LG4:=1300000;
P1:OBJ:=P1,LT1:=605000,LG1:=1474500,LT2:=545000,LG2:=1481500,
            LT3:=615500,LG3:=1533370,LT4:=55....,LG4:=154120,
            LT5:=581000,LG5:=1510000,ROW:=775,COL:=900,
    FORMAT:=FORMATB,VOL:='SER=TAPE1',FILE:=3,ID:=TAPE1;
P2:OBJ:=P2,LT1:=544000,LG1:=1541000,LT2:=535400,LG2:=1545000,
            LT3:=555200,LG3:=1595200,LT4:=535400,LG4:=1555000,
            LT5:=520600,LG5:=1573500,ROW:=775,COL:=900,
    FORMAT:=FORMATB,VOL:='SER=TAPE2',FILE:=6,ID:=TAPE2;
P3:OBJ:=P3,LT1:=432800,LG1:=1611500,LT2:=422200,LG2:=1623000,
            LT3:=494000,LG3:=1673300,LT4:=425730,LG4:=1633570,
            LT5:=460900,LG5:=1645900,ROW:=775,COL:=900,
    FORMAT:=FORMATB,VOL:='SER=TAPE3',FILE:=5,ID:=TAPE3;
P4:OBJ:=P4,LT1:=415100,LG1:=1691000,LT2:=361500,LG2:=1705000,
            LT3:=440900,LG3:=1750100,LT4:=374000,LG4:=1770500,
            LT5:=401900,LG5:=1733800,ROW:=775,COL:=900,
    FORMAT:=FORMATB,VOL:='SER=TAPE4',FILE:=6,ID:=TAPE4;
P5:OBJ:=P5,LT1:=310900,LG1:=1870300,LT2:=253200,LG2:=1890300,
            LT3:=334800,LG3:=1931400,LT4:=270000,LG4:=1953130,
            LT5:=292900,LG5:=1912900,ROW:=775,COL:=900,
    FORMAT:=FORMATB,VOL:='SER=TAPE5',FILE:=4,ID:=TAPE5;
P6:OBJ:=P6,LT1:=520600,LG1:=1821300,LT2:=462100,LG2:=1834500,
            LT3:=541100,LG3:=1880130,LT4:=465900,LG4:=1894500,
            LT5:=500400,LG5:=1855900,ROW:=775,COL:=900,
    FORMAT:=FORMATB,VOL:='SER=TAPE6',FILE:=3,ID:=TAPE6;
P7:OBJ:=P7,LT1:=632800,LG1:=1880900,LT2:=574200,LG2:=1895100,
            LT3:=655600,LG3:=1932700,LT4:=585100,LG4:=1950500,
            LT5:=614900,LG5:=1915800,ROW:=775,COL:=900,
```

24

```
          FORMAT:=FORMATB,VOL:='SER=TAPE7',FILE:=2,ID:=TAPE7;
R4:(EXTENT COORD LT2<LT4<EXTENT COORD LT1) AND
    (EXTENT COORD LG2<LG4<EXTENT COORD LG3);
R5:(EXTENT COORD LT2<LT5<EXTENT COORD LT1) AND
    (EXTENT COORD LG2<LG5<EXTENT COORD LG3);
REGIONA:R1 OR R2 OR R3 OR R4 OR R5;
EXTENT:OBJ:=EXTENT,COORD:=PROJ1;
VSETUP:VICRUNX,JOBLIB,STEP1,SYSOUT,SYSUDUMP,FT06F001,FT07F001,
         VSYS00;
VICRUNX:JCL:=JOB,BLANK:=(SYSTEM,USER,JOBID),BLANK:=(
        SYSTEM,USER),CLASS:=D,TIME:=15;
JOBLIB:JCL:=DD,DSN:='IPL1.VICARLIB',DISP:=SHR,VOL:='SER=000530',
        UNIT:=2314;
STEP1:JCL:=EXEC,PGM:=VMAST,REGION:='250K',TIME:=15;
SYSOUT:JCL:=DD,SYSOUT:='A',DCB:='BLKSIZE=3520';
SYSUDUMP:JCL:=DD,SYSOUT:='A';
FT06F001:JCL:=DD,SYSOUT:='A';
FT07F001:JCL:=DD,SYSOUT:='B';
VSYS00:JCL:=DD,UNIT:=2314,SPACE:='(TRK,2)';
SYSIN:JCL:=DD,BLANK:='*',DCB:='BLKSIZE=80';
MOSAIC:VICAR:=MOSAIC,JCL:=VSETUP,PGM:=(PROJECT,MOSAIC2),OUTPUT:=
              (SYSTEM,USER,OUTNAME);
PROJECT:PGM:=PROJECT,INPUT:='@',OUTPUT:=TEMP,PARAMS:=(PARAM1,
       PARAM2),REPEAT:='@',OUTSIZE:=(1,1,2000,2000);
PARAM1:PARAMS:=(EXTENT,COORD);  PARAM2:PARAMS:=('@',COORD);
MOSAIC2:PGM:=MOSAIC2,INPUT:=(TEMP),OUTPUT:=(SYSTEM,USER,OUTNAME);
FORMATA:JCL:=DD,UNIT:=2314,DCB:='(BLKSIZE=6000,LRECL=2000)',
        SPACE:='(6000,(0710),,,ROUND)';
FORMATB:JCL:=DD,LABEL:='(,BLP)',DISP:='(OLD,KEEP)',UNIT:=TAPE7,
        DCB:='(BLKSIZE=0,LRECL=0,DEN=2,TRTCH=C)',VOL:=('@',VOL);
NIMENSKY:JOBID:='(08648,98312,3325)',OUTNAME:=BOBOUT;
BOBOUT:FORMAT:=FORMATB,VOL:='SER=BOBOUT',ID:=BOBOUT;
SYSTEM:USER:=NIMENSKY;
#MOSAIC TEST 4/18/73;
 TERMINAL
```

APPENDIX B

SAMPLE CREATED JOB